ELSEVIER International Journal of Approximate Reasoning 23 (2000) 85–109

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING

www.elsevier.com/locate/ijar

GA-based learning for a model-based object recognition system

R. Soodamani, Z.Q. Liu *

Computer Vision and Machine Intelligence Laboratory (CVMIL), Department of Computer Science, The University of Melbourne, 221 Bouverie Street, Carlton, Victoria 3053, Australia

Received 1 December 1997; received in revised form 1 December 1998; accepted 1 February 1999

Abstract

This paper proposes a genetic-algorithm-based learning strategy that models membership functions of the fuzzy attributes of surfaces in a model based machine vision system. The objective function aims at enhancing recognition performance in terms of maximizing the degree of discrimination among classes. As a result, the accuracy of recognizing known instances of objects and generalization capability by recognizing unknown instances of known objects are greatly improved. Performance enhancement is achieved by incorporating an off-line learning mechanism using genetic algorithm in the feedback path of the recognition system. © 2000 Elsevier Science Inc. All rights reserved.

Keywords: Model based object recognition; Range images; Fuzzy attributes; Learning; Genetic algorithm; Performance enhancement

1. Introduction

The need for learning in object recognition systems arises from the fact that they may exhibit suboptimal performance due to the lack of the experience gained at one or more stages such as data acquisition, concept representation and/or hypotheses generation. This is quite often the case with fuzzy systems,

* Corresponding author. Tel.: +61-3-9344-9124; fax: +61-3-9344-1184.
E-mail addresses: sor@krang.vis.mu.oz.au (R. Soodamani); zliu@cs.mu.oz.au, zliu@splinter.vis.mu.oz.au (Z.Q. Liu).

PII: S0888-613X(99)00036-5

where certain stages of the system design are initiated from expert's knowledge and later fine tuned on a trial-and-error basis based on the performance of the system. Most vision systems employ a top–down feedback for dynamically exploiting low-level image data. However, because a process that includes low-level image processing in the scope of control is complex, most control structures are restricted at the symbolic level and only a few have feedback to the image level. A realistic way to overcome this drawback is by using learning methods that automatically select the best parameter values in a specific task.

Although vision systems employ some form of feedback control mechanism, little research has been done [1–3]. At present, standard optimization techniques cannot correctly and adequately handle the complexity of vision problems. Parameter tuning remains to be a bottleneck for the development of evolutionary vision systems for two main reasons: First, the search spaces for vision problems are multi modal, a property that results in local minima; Second, for some applications, the search space is discontinuous which makes it difficult to implement hill-climbing techniques. Genetic algorithms (GAs) are able to avoid these problems.

The GAs have been employed for generating fuzzy if-then rules and adjusting membership functions of fuzzy sets [4–7]. Systems that automatically learn classification rules are found in [8–12]. In [8], a fuzzy classifier system is proposed in which each fuzzy rule is encoded as an individual in the genetic algorithm. The fuzzy subsets in the antecedent and consequent parts of a rule are automatically selected by the system. In [9] an evolutionary learning mechanism is proposed for morphological pattern recognition. Such a technique is suitable for morphology-based recognition systems that require the analysis of the geometrical and topological properties of images. The technique proposed in [11] is an extension of [10]. In this paper, fuzzy rules with variable fuzzy regions are defined by activation *hyperboxes* which show the existence regions of the data for that class. These rules are extracted from numerical data by recursively resolving overlaps between two classes. The optimal input variables for the rules are determined by using the number of extracted rules as a criterion. In [12], a GA-based method for classification is proposed for selecting a small number of significant fuzzy if-then rules. They formulate the rule selection problem as a combinatorial optimization problem with two objectives: maximizing correctly classified patterns and minimizing the number of fuzzy if-then rules. A set of fuzzy if-then rules is encoded into a string and treated as an individual in genetic algorithms. Both methods [11,12] use the Fisher data [13] in their experiments and test for validation and generalization.

In this paper, we consider a model-based object recognition system consisting of a set of synthetic range images whose canonical views are used as instances in an incremental model-building process. These images pass through a segmentation module [14] which partitions the object into surfaces that satisfy certain smoothness criteria. In the next stage, we extract a set of features

from these surfaces and initially model the features in terms of fuzzy membership functions based on a priori knowledge of the feature parameters. We represent each model based on these surface features and match test objects. Each of the modules influences the overall performance of the object recognition system. The aim of this work is to improve the performance by tuning the parameters of a specific module, namely, feature fuzzification.

A GA-based learning process is incorporated in the feedback path of the system that connects the recognition stage to the input stage where fuzzy modeling of the features takes place. The fuzzy rules, the shape of the membership functions, and the extent of overlaps between the functions are learnt dynamically. The objective function aims at maximizing discrimination between the object models, thereby minimizing misclassification. Learning is done off-line and is terminated when satisfactory performance is reached. We show that the proposed learning process improves the recognition performance of the system under validation and generalization tests. In addition, the performance of such a system is tested for rejection hypothesis of alien objects to the model set as well as for its robustness to various levels of noise in the test images during validation.

The rest of the paper is organized as follows: in Section 2, the general system architecture of the fuzzy object recognition system requiring learning is discussed. In Section 3, the performance evaluation criteria for the above system is discussed. In Section 4, we define the objective function for the recognition system and introduce the concept of the dynamic learning mechanism. Section 5 outlines the design and specifications of the GA-based learning paradigm, which is incorporated into the system to improve the performance. In Section 6, we discuss the performance improvement due to learning.

2. Object recognition system architecture

In this section, we discuss the general system architecture of object recognition system (ORS), concentrating on the feature representation and fuzzification, which directly affect the performance of the system. Fig. 1 outlines the various modules involved in the object recognition system. The modules within dotted lines indicate the data structure of the modules they are associated with. There are two parts to this system: a modeling phase and a testing phase. During modeling, the Incremental Model Construction module is included, while during testing, this phase is not necessary and a diversion takes place. The details of the individual modules are described in the following sections.

2.1. Input images

The object database has synthetic range images of five chess pieces each of which has five different instances [15]. This database is obtained from the

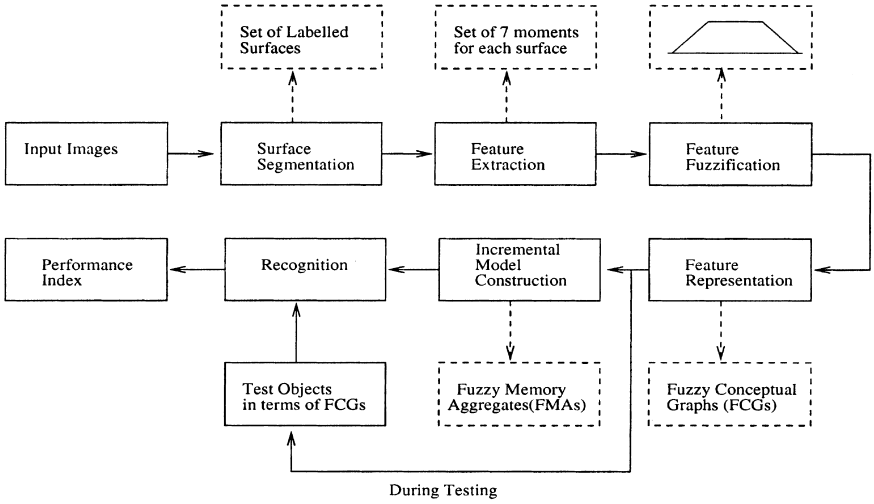

Fig. 1. General system architecture of ORS.

Computer Vision and Machine Intelligence Laboratory, Department of Computer Science, The University of Melbourne, Australia. These objects are complex in that they bear close resemblance to each other and require sensitive treatment of parameter fuzzification to provide enough discrimination between them. This set is suitable for testing in a controlled environment, wherein different levels of noise can be added into the images and the resulting images can be used for robustness tests.

2.2. Surface segmentation and feature extraction

The input images of objects require preprocessing for deriving a suitable description of the objects. This consists of a segmentation process based on curvature measures [16–18] by which an object is segmented into surface patches of a specific type. Typically, they may be classified as one of the following: pit, saddle valley, valley surface, saddle ridge, ridge, peak, minimum surface, or flat surface. Surface-based descriptions are then derived to form a feature vector for each object. A set of seven central moments [19] are used to generate shape descriptors on each surface shape. In this paper, for an input image, the result of segmentation is a set of sub-images corresponding to each surface type. The moment invariants are estimated for each labeled image, for all instances of the object. The dimension of the resulting feature vector is determined by the number of instances of the object, the number of possible surface types, and the seven invariant moments. This converts the pattern

recognition problem into a standard decision theory problem, to which several approaches are available.

2.3. Feature fuzzification

The set of invariant moments are modeled by fuzzy parametric functions within their overall universe of discourse. A set of five trapezoidal membership functions is used to model the set of seven invariant moments. This part of the modeling influences the discriminating ability between objects within the class. An initial heuristic set of parameters is derived allowing a 10–25% overlap within the support region of each membership function. The fuzzification of the input space is carried out using the membership functions shown in Fig. 2 and defined as follows:

$$\mu_A(x) = \begin{cases} 1 & \text{if } x \leqslant c, \\ (d-x)/(d-c) & \text{if } c < x \leqslant d, \end{cases} \tag{1}$$

$$\mu_B(x) = \begin{cases} 0 & \text{if } x \leqslant a, \\ (x-a)/(b-a) & \text{if } a < x \leqslant b, \\ 1 & \text{if } b \leqslant x \leqslant c, \\ (d-x)/(d-c) & \text{if } c < x \leqslant d, \\ 0 & \text{if } x > d, \end{cases} \tag{2}$$

$$\mu_C(x) = \begin{cases} (x-a)/(b-a) & \text{if } a < x \leqslant b, \\ 1 & \text{if } x \leqslant x \geqslant b. \end{cases} \tag{3}$$

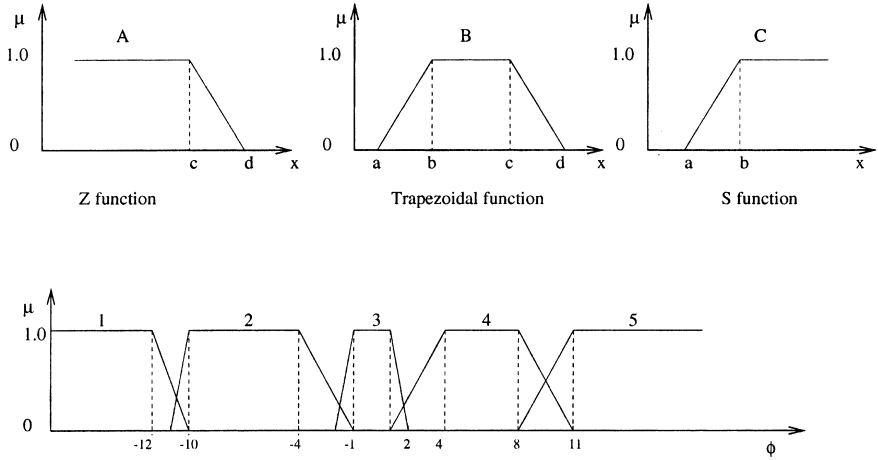

Fig. 2. Membership functions and input space division for ORS.

Eqs. (1)–(3) are defined as the z-, trapezoidal- and s-membership functions. In Fig. 2 is also shown a sample of the input space (moment) division for ORS.

2.4. Feature representation fuzzy conceptual graphs

The set of fuzzy moments derived as above needs to be represented in a form suitable for machine learning. Conceptual graphs [20] can model facts that can be subjected to generalized reasoning. Such graphs maintain a hierarchy of concept-relation structures and can *create* new instances, *join* instances to form a memory aggregate (MA) and *match* a query (test object) against the MA. In order to provide a means for evidential reasoning under uncertainty [21], we modify the conceptual graphs to incorporate fuzzy features in their structures. We call these graphs fuzzy conceptual graphs (FCGs). The hierarchical arrangement of FCG for the considered object recognition system is shown in Fig. 3.

2.5. Fuzzy memory aggregates

Having represented the features through FCGs, we derive from its instances the fuzzy memory aggregate (FMA) that is a compact model of the object. The FMA takes the form of an FCG. Initially, the FMA has null features in it. As each instance is submitted to the system the FMA builds up a model incrementally by an *OR* operation between the instance and the FMA. As shown in Fig. 3, the model is constructed at Level 4, where the set of nodes forms a fuzzy feature vector of moments for a specific surface type. If ϕ_{kqrmp} represents the pth fuzzy attribute of mth invariant moment and r is a specific surface type, then for a set of five instances of object k, the following set of data forms ϕ_{kqrmp},

$$\phi_{k1rm} = (0.000000\ 0.271091\ 0.186726\ 0.000000\ 0.000000), \tag{4}$$

$$\phi_{k2rm} = (0.000000\ 0.558913\ 0.000000\ 0.000000\ 0.000000), \tag{5}$$

$$\phi_{k3rm} = (0.000000\ 1.000000\ 0.000000\ 0.000000\ 0.000000), \tag{6}$$

$$\phi_{k4rm} = (0.000000\ 0.513543\ 0.000000\ 0.000000\ 0.000000), \tag{7}$$

$$\phi_{k5rm} = (0.000000\ 1.000000\ 0.000000\ 0.000000\ 0.000000). \tag{8}$$

The resulting FMA at this specific level is the average of the corresponding column values of Eq. (4) through (8). Care is taken to ensure that varying the order of presenting the instances does not result in different FMAs being constructed.

$$\phi_{krm} = (0.000000\ 0.668709\ 0.0373452\ 0.000000\ 0.000000). \tag{9}$$

In order to obtain a complete model for object k, we repeat this process for each surface type (r) and each moment (m), over all views. In terms of image processing, each model of an object is a feature vector of dimension given by

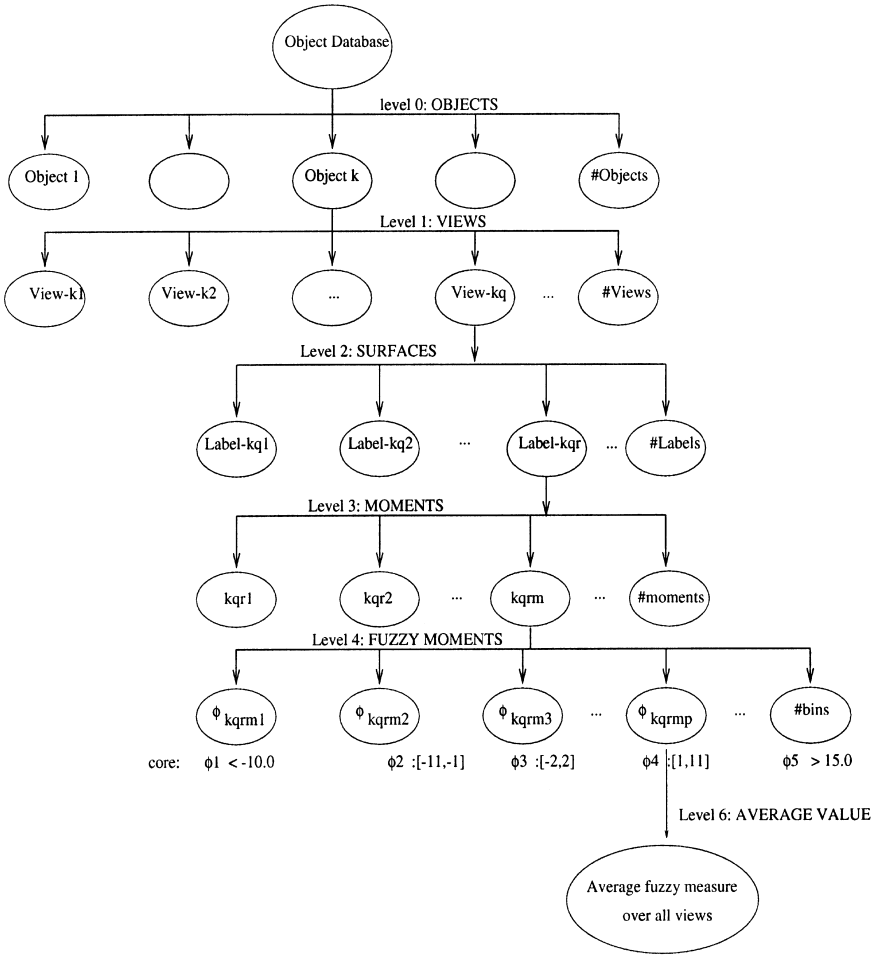

Fig. 3. FCG of ORS.

(size of r) $\times$ (size of m) $\times$ (size of p), where p indicates a fuzzy attribute. The resulting FMA of an object is a hierarchical structure of a set of aggregated fuzzy invariant moments associated with each surface patch.

2.6. Recognition by projection

With FCGs, reasoning is performed by projecting observed features onto FMAs. A test object, called a query, is first represented as an FCG for performing matching with the FMA. A necessity measure $\mathcal{N}$ is derived when corresponding nodes within graphs match. Only the information that is both in

the FMA and the query will contribute to the increase of $\mathscr{N}$ that is derived by a *min* operation over the fuzzy measures of the query and the model. Information that is in the FMA but not in the query or vice-versa is treated as conflicting information and contributes to decreasing the possibility measure. The disbelief $\mathscr{P}$ is derived by a *max* operator over the fuzzy measures of the FMA and the query (when they conflict). $\mathscr{N}$ and $\mathscr{P}$ act as consonant and dissonant measures of evidence, respectively [22]. The certainty of match is given by the difference of the $\mathscr{P}$ and $\mathscr{N}$ measures. The difference ensures a lower bound on the certainty of match. That is, the system is confident of the extent to which recognition can take place. If Q is a query, then the following algorithm gives the steps involved in determining the degree of match (or certainty) $\mathscr{C}_{Qk}$ between Q and FMA corresponding to each object k. The necessity and possibility measures of Q versus FMA of k are given by $\mathscr{N}_{Qk}$ and $\mathscr{P}_{Qk}$.

Initialize

$$\mathscr{N}_{Qk} = 0;\ \mathscr{P}_{Qk} = 0;\ \mathscr{C}_{\max} = 0;\ d_{Qk} = 0;\ \mathscr{C}_{Qk} = 0.$$

$\forall r, m, p$ if $|\phi_{rmp}(Q)| > 0,$ or $|\phi_{rmp}(\mathrm{FMA})| > 0,$

$$\begin{aligned}
&\mathscr{N}_{Qk} = \mathscr{N}_{Qk} + \min_p(\phi_{rmp}(Q),\ \phi_{rmp}(\mathrm{FMA})) && \text{if } \phi_{rmp}(Q) < \phi_{rmp}(\mathrm{FMA}),\\
&\mathscr{P}_{Qk} = \mathscr{P}_{Qk} + \max_p(\phi_{rmp}(Q),\ \phi_{rmp}(\mathrm{FMA})) && \text{if } \phi_{rmp}(Q) > \phi_{rmp}(\mathrm{FMA}),\\
&\mathscr{C}_{Qk} = \mathscr{C}_{Qk} + \mathscr{N}_{Qk} - \mathscr{P}_{Qk}, &&\\
&\mathscr{C}_{\max} = \mathscr{C}_{Qk} && \text{if } \mathscr{C}_{Qk} > \mathscr{C}_{\max}.
\end{aligned} \tag{10}$$

Based on $\mathscr{C}_{Qk}$ and $\mathscr{C}_{\max}$, a distance metric d_{Qk} is derived that determines the match score normalized within [0,1] as defined by Eq. (12). A match is said to occur if the score value exceeds a threshold. The value of this threshold is set to 0.9 for ORS. i.e., Q matches FMA if $\mathscr{M}_{Qk} > \mathit{threshold}$.

$$d_{Qk} = \begin{cases} \mathscr{C}_{\max} - \mathscr{C}_{Qk} & \text{if } \mathscr{C}_{Qk} > 0,\\ \mathscr{C}_{Qk} & \text{otherwise,} \end{cases} \tag{11}$$

$$\mathscr{M}_{Qk} = 1.0/(1.0 + d_{Qk}). \tag{12}$$

Note that there might be a set of objects for which this match can take place. By tuning the threshold value, the set may be reduced to a single object match. It is difficult to illustrate this evidence accumulation process with an example. Partial steps of projection and evidence accumulation are nevertheless illustrated as follows. Let the following set of data represent models such as that in Eq. (9), corresponding to a five class object recognition system,

$$\phi_{1rm} = (0.000000\ 0.668709\ 0.0373452\ 0.000000\ 0.000000), \tag{13}$$

$$\phi_{2rm} = (0.000000\ 0.726453\ 0.0000000\ 0.000000\ 0.000000), \tag{14}$$

$$\phi_{3rm} = (0.000000\ 0.700989\ 0.0000000\ 0.000000\ 0.000000), \tag{15}$$

$$\phi_{4rm} = (0.000000\ 0.833333\ 0.0000000\ 0.000000\ 0.000000), \tag{16}$$

$$\phi_{5rm} = (0.000000\ 0.597527\ 0.0000000\ 0.000000\ 0.000000). \tag{17}$$

Let Eq. (4) be the query Q. This implies that the best score of match must be $\mathscr{M}_{Q1}$. Now $Q = \phi_{Qrmp}$. Projection of ϕ_{Qrmp} on ϕ_{1rmp} results in the following:

$$\begin{aligned} \mathscr{N}_{Qk} &= 0.000000 + 0.271091 + 0.000000 + 0.000000 + 0.000000 \\ &= 0.271091, \\ \mathscr{P}_{Qk} &= 0.000000 + 0.000000 + 0.186726 + 0.000000 + 0.000000 \\ &= 0.186726, \end{aligned} \tag{18}$$

$$\mathscr{C}_{Q1} = \mathscr{N}_{Qk} - \mathscr{P}_{Qk} = 0.271091 - 0.186726 = 0.084265. \tag{19}$$

Similarly, projection of ϕ_{Qrmp} on ϕ_{krmp} for $k = 2, \ldots, 5$ results in

$$\mathscr{N}_{Q2} = \mathscr{N}_{Q3} = \mathscr{N}_{Q4} = \mathscr{N}_{Q5} = 0.271091, \tag{20}$$

$$\mathscr{P}_{Q2} = \mathscr{P}_{Q3} = \mathscr{P}_{Q4} = \mathscr{P}_{Q5} = 0.000000, \tag{21}$$

$$\mathscr{C}_{Q2} = \mathscr{C}_{Q3} = \mathscr{C}_{Q4} = \mathscr{C}_{Q5} = 0.271091. \tag{22}$$

Repeating this process for all r, m, p, k, let us assume the following cumulative result:

$$\begin{aligned} \mathscr{C}_{Q1} &= 14.187467, \\ \mathscr{C}_{Q2} &= 2.765325, \\ \mathscr{C}_{Q3} &= 5.748958, \\ \mathscr{C}_{Q4} &= -1.205288, \\ \mathscr{C}_{Q5} &= 8.713797, \end{aligned} \tag{23}$$

from which $Cert_{\max} = 14.187467$. Employing Eqs. (11) and (12), the distance metric and correlation scores are derived as shown in Table 1.

In Table 1, the asterisk * indicates that $\mathscr{C}_{Q4}$ is negative, implying a negative correlation factor. Hence no further calculation is performed on negative certainty factors except assigning that value itself as a score of match. From this table, it is evident that the best matching score is obtained for Class 1 with the utmost certainty and is considered the correct match.

Table 1
Match scores vs. distance metric

Class	d_{Qk}	$\mathscr{M}_{Qk}$
1	0.000000	1.000000
2	11.422142	0.080501
3	8.438509	0.105949
4	*******	−1.205288
5	5.47367	0.154472

3. Performance evaluation of ORS

Performance evaluation of ORS is carried out with the set of objects described in Section 2.1. The objects in the database are as shown in Fig. 12. We conducted four different types of tests using this database which are discussed in the following sections.

3.1. Validation test

This test evaluates recognition performance with known instances of the objects. These instances were used originally for constructing the model base. All instances were tested. We therefore had 25 test instances matched against five classes of objects. In this test, the system showed its ability to recognize the correct instances corresponding to an object model and reject instances corresponding to the rest of the models within the same database.

3.2. Generalization test

This test evaluates recognition performance with unknown instances of the known objects. Given a model, the system is tested for its ability to recognize intermediate views of an object not seen before. To achieve this, the original memory aggregates (models) were constructed with a *leave-one-out* strategy and the left-out object is included in the test. At a time, only 4 instances of each object are considered for model aggregation. This implies that the partial model has only 80% of the complete model used in the validation test, but the entire object database is used for testing, that is 25. With this strategy, five different experiments are conducted, wherein a specific view (instance number) is left out for each of the objects. For example, view number 1 is left out of all objects in the first experiment, view 2 in the second, etc. and view 5 in the last experiment. The test results show that the system is able to derive good generalizations from examples and to recognize unseen instances of existing models.

3.3. Robustness to noise test

This test evaluates recognition performance with known instances of the objects with the test objects being subjected to varying noise levels. White Gaussian noise is added to the test images having the following signal-to-noise-ratios (SNR) 1.0, 1.5, 10, 50, 100, and 200, where the SNR is defined as the ratio of signal power σ_s^2 to noise power σ_n^2, and the signal power is assumed to be unity,

$$\mathrm{SNR} = \frac{\sigma_s^2}{\sigma_n^2}.$$

In this test, only one instance of each class is considered. For each instance, six different noise levels were used. That is, there are five test objects of the same SNR, resulting in $5 \times 6 = 30$ test objects. Recognition performance is classified according to the SNR. Furthermore, a combined test involves a mixture of all the noisy images, which determines the overall performance for the noisy images.

3.4. Rejection hypothesis

This test evaluates the system's ability to reject objects that are *alien* to its model base. For this purpose, we used a second set of database consisting of real range images. The model base has 11 objects each of which has five instances as shown in Figs. 13 and 14. These object images were obtained from Pattern Recognition and Image Processing Laboratory, Michigan State University, USA.

Table 2 gives the performance results of ORS using the chess database. The validation performance is 72%. With the leave-one-out strategy, the performance varies between 60–72%. The rejection performance is 84%. The table also shows recognition performance with noisy test images. Columns 2–6 indicate the performance of test objects along with their corresponding SNRs. The poor performance has no direct bearing on the level of noise in the image. For instance, the relative performance with $\mathrm{SNR} = 1.0$ is quite high in contrast to an unexpected poor performance at $\mathrm{SNR} = 100$. The last column gives the performance measure when the whole set of noisy objects including all levels of noise are tested which is slightly higher than the individual tests.

Table 2
Performance evaluation of chess model base

Validation performance		Generalisation performance						
Recognition %	Rejection %	Model	Recognition %	Rejection %				
72	84	1	72	84				
		2	64	83				
		3	60	84				
		4	64	83				
		5	72	84				
Validation performance with noisy test objects								
SNR	1.0	1.5	10	50	100	200	Mixed	
Recognition %	40	20	40	40	20	40	43	

4. Performance enhancement by genetic algorithm

To improve the performance of the system, ORS can be segregated into two parts: the static part consisting of the modules of surface segmentation and feature extraction and the dynamic part that has feature fuzzification, representation, model construction, and recognition. These modules are considered dynamic because any change in feature fuzzification module causes a dynamic effect on each of them. We optimize the performance by tuning the following parameters: the number of fuzzy partitions P in the input space, the extent of overlaps between these partitions, and the shape of the membership functions governing each partition. These parameters are tuned by using a GA-based learning mechanism within the dynamic part of the system as shown in Fig. 4. The objective function aims at minimizing misclassification as follows:

$$\text{Minimize}\left(\sum_{\forall Q}\sum_{\forall k, Q\neq k}\mathscr{M}_{Qk}/N' - \sum_{\forall Q}\sum_{\forall k, Q=k}\mathscr{M}_{Qk}/N''\right), \tag{24}$$

$$N' = \sum_{\forall Q}\sum_{\forall k} n'(Q,k), \quad N'' = \sum_{\forall Q}\sum_{\forall k} n''(Q,k), \tag{25}$$

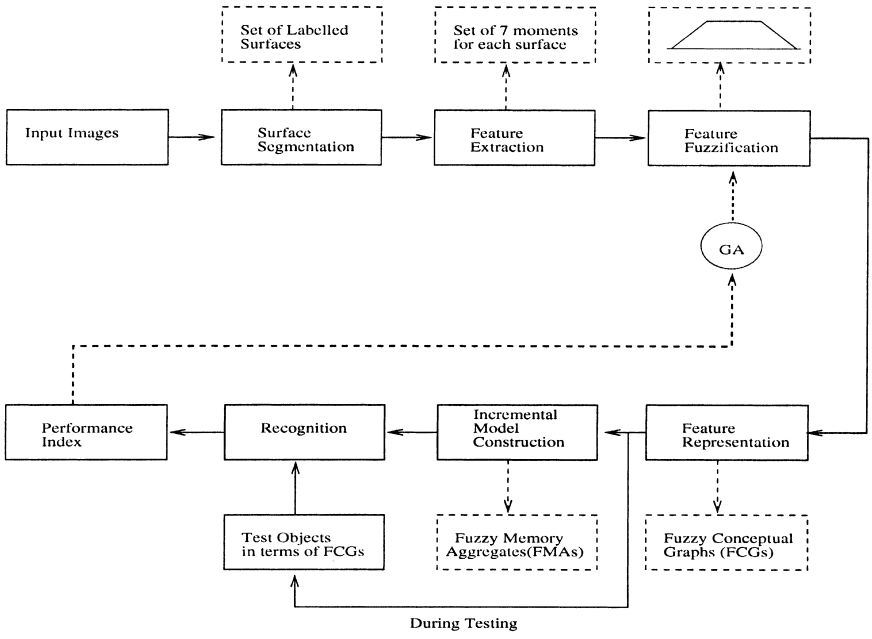

Fig. 4. Learning in ORS.

$$n'(Q,k) = \begin{cases} 1 & \text{if } Q \neq k, \\ 0 & \text{otherwise,} \end{cases} \tag{26}$$

$$n''(Q,k) = \begin{cases} 1 & \text{if } Q = k, \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

Eq. (24) consists of two terms: the first term is associated with the degree of false matches and the second term is associated with the degree of correct matches. The optimization function decreases for smaller measures of false matches in relation to that of correct matches. The higher the disparity between these two terms the better is the discrimination between object classes. Eqs. (26) and (27) indicate the total number of false and correct matches during recognition. Thus the optimization function in Eq. (24) is the fitness function of a genetic algorithm.

The first iteration of learning starts with parameters as defined in Fig. 2. That is, the learning process is initialized with the parameters of the original ORS that does not have the learning capability. Initially, the number of fuzzy partitions P is 5, the extent of overlap is set to vary between 10% and 25% of the span of each membership function $\mu_i(\phi)$, and their shapes are in general trapezoidal. In this paper, once the learning mechanism is incorporated into the system, the shapes of $\mu_i(\phi)$ take highly irregular forms. Therefore, the initialization process requires the values of $\mu_i(\phi)$ at finely sampled points along the input co-ordinate. This process approximates the trapezoidal membership functions only in the first iteration. Each feasible solution of the optimization function is treated as an individual in genetic algorithms. The rule set is represented as a string.

4.1. Extent of overlaps

Depending on the value of P, the universe of discourse U is divided into equal partitions as shown in Fig. 5 by dotted lines at points $C_{12}, C_{23}, \ldots$ The end points (a_i, d_i) of each $\mu_i(\phi)$ is placed about C_{i-1i}, C_{ii+1} such that $a_i \leqslant C_{i-1i}$ and $d_i \geqslant C_{ii+1}$ satisfy the conditions $(C_{i-1i} - a_i)/S \leqslant 1.0$ and $(d_i - C_{ii+1})/S \leqslant 1.0$, respectively, where S is the equidistant span along ϕ. This in effect varies the extent of overlap between $\mu_i(\phi)$ as a fraction of the span.

4.2. Sample points

The sampling rate N_i is maintained a constant for each fuzzy partition P_i. Given (a_i, d_i) of $\mu_i(\phi)$, we can determine each sample value ϕ_{ij} as follows:

$$\phi_{ij} = a_i + (D_i/N_i) * j, \tag{28}$$

$$D_i = d_i - a_i, \tag{29}$$

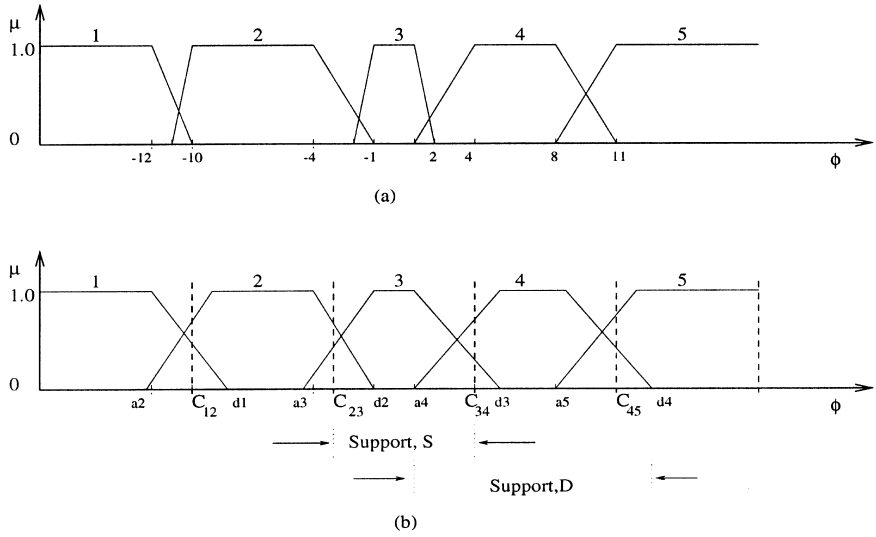

Fig. 5. Designing fuzzy partitions and overlaps.

where D_i is the span(support) of P_i. We assume that $\mu_i(\phi_{ij})$ is available. Since the sampling rate is fixed for all fuzzy subsets, some of these subsets can be more finely sampled than the others.

4.3. Membership values at sample points

Since the sampling rate is high, we may consider $\mu_i(\phi)$ to be piecewise linear, without any loss of generality. For any unknown input ϕ', $\mu_i(\phi')$ can be determined using the slope-intercept form of linearity,

$$\mu_i(\phi') = (\mu_i(\phi_2) - \mu_i(\phi_1))/(\phi_2 - \phi_1), \tag{30}$$

$$\phi_1 \leqslant \phi' \leqslant \phi_2, \quad \phi_2 \neq \phi_1, \tag{31}$$

where ϕ_1, ϕ', ϕ_2 are consecutive values of ϕ.zz

5. Genetic coding and learning

GA's are iterative procedures which maintain a *population* of candidate solutions to the objective function $f(x)$,

$$P(t) = \{x_1(t), x_2(t), \ldots, x_N(t)\}. \tag{32}$$

Each structure x_i in P is simply a binary string of length L. Generally, each x_i represents a vector of parameters to the function $f(x)$, but the semantics associated with the vector is unknown to the GA. During each iteration step,

called a *generation*, the current population is evaluated, and, on the basis of the evaluation, a new population of candidate solutions is formed. For a general overview of genetic algorithms please refer to [5]. The basic concepts of GA's were developed by Holland [4] and his students [23–26]. A number of experimental studies [4,23,24,27] have shown that in practice, GA's exhibit impressive efficiency for heuristic information gathering in complex search spaces. While classical gradient search techniques are more efficient for problems that satisfy tight constraints (e.g., continuity, low dimensionality, unimodality, etc.), GA's consistently outperform both gradient techniques and various forms of random search on more difficult (and more common) problems, such as optimizations involving discontinuous, noisy, high dimensional, and multimodal objective functions. GA's have been applied to various domains, including numerical function optimization [23,27], adaptive control system design [6,7,24], artificial intelligence task domains [28] and pattern classification [5,9,12]. In the following sections, we discuss the design of the learning module for dynamically adapting the feature fuzzification process as a function of performance.

The process of learning the fuzzy parameters of ϕ is achieved by encoding P, (a_i, d_i) for each P_i as a percentage variation within D, and the pair of values $\phi_{ij}, \mu_i(\phi_{ij})$ as a string in GA. We use the genetic code GENESIS Version 5.0 for implementing the learning process. GENESIS is capable of handling floating point representations of genetic structures. One *generation* of GA comprises of the steps of selection, mutation, crossover, evaluation, and some data collection procedures. These processes and the initialization procedure are discussed in the following sections.

5.1. Initialization

Generate an initial population containing *Popsize* strings where *Popsize* is the number of strings in each population. In this operation, GENESIS requires the length of the string as the number of genes in *Popsize* and the description of an individual or a set of genes in terms of the range of values they take and the repetition count if descriptions are similar. In this experiment, we assume that the maximum number of fuzzy partitions is 10. Hence the data structure corresponding to this size is necessary to be provided to GENESIS. Thus the first gene specifies P and is allowed to vary between 5 and 10. Based on P, the data structure is read for $(a_i, d_i) \in [0, 1]$ and $\phi_{ij}, \mu(\phi_{ij}) \in [0, 1]$. For convenience, $\mu(\phi_{ij})$ is written as μ_{ij}. The resulting individual in a population appears as follows:

$$\{P, \{(a_1, d_1), (a_2, d_2), \cdots, (a_P, d_P)\}, \{\mu_{11}, \mu_{21}, \ldots, \mu_{N1}\}, \{\mu_{12}, \mu_{22}, \ldots, \mu_{N2}\} , \ldots, \{\mu_{1P}, \mu_{2P}, \ldots, \mu_{NP}\}\}, \tag{33}$$

where N is the number of sample points within each fuzzy bin. In the above structure, P is a variable depending on which the length of the structure varies. With GENESIS, the maximum and minimum values of each element of the encoded string need to be specified. Thus $5 \leqslant P \leqslant 10$, $0 \leqslant a_P$, $d_P \leqslant 1.0$, $0 \leqslant \mu_{ij} \leqslant 1.0$ and $N = 101$. a_P, d_P are specified as a proportion of the span of fuzzy subsets along ϕ.

5.2. Selection

Selections is a stochastic procedure that guarantees the number of offspring of any structure to be bounded by the floor and by the ceiling of the (real-valued) expected number of offspring. This procedure is based on an algorithm by James Baker [29]. The idea is to allocate to each structure a portion of a spinning wheel proportional to the structure's relative fitness. A single spin of the wheel determines the number of offspring assigned to every structure. The structures are copied into the new population.

Selection may also be based on a ranking algorithm in which the probability of selecting a structure is proportional to its index in the population. Ranking helps prevent premature convergence by preventing too fit individuals from taking over the population within a few generations. However, ranking also produces slower improvement over proportional selection.

5.3. Mutation

After the new population is selected, mutation is applied to each structure in the new population. Each position is given a chance of undergoing mutation. This is implemented by computing an interarrival interval between mutations, assuming a mutation rate of M_{rate}. If mutation does occur, a random value is chosen from $\{0, 1\}$ for that position. If the mutation differs from the original structure, the structure is marked for evaluation.

5.4. Crossover

Crossover exchanges alleles among adjacent pairs of the first structures in the new population. The segments between the crossover points are exchanged, provided that the parents differ somewhere outside of the crossed segment. If, after crossover, the offspring are different from the parents, then the offspring replace the parents, and are marked for evaluation.

5.5. The elitist strategy

In this paper, the "Elitist" selection strategy is opted which stipulates that the best performing structure always survives intact from one generation to the

next. In the absence of this strategy, it is possible that the best structure disappears due to crossover or mutation. This is achieved by randomly removing one string from *Popsize* strings generated by the above operations, and adding the best string with the maximum fitness in the previous population to the current one.

5.6. Termination test

Repeat all of the above steps until the specified number of generations is reached.

In this paper, $Popsize = 100, M_{\text{rate}} = 0.001$, crossover rate $C_{\text{rate}} = 0.6$ and number of generations varies for different experiments. The structure length is 5160 for $P = 5$ and 10320 for $P = 10$. The floating point representation of (a_i, d_i) and μ_{ij} are represented by 10 bit binary strings. For $P = 5$, the number of parameters involved in the structure specified by Eq. (33) is given by $1 + (5 \times 2) + (5 \times 101) = 516$. The corresponding search space has dimensionality $1 + 10 \times 2^{10} + 505 \times 2^{10} \approx 2^{20}$. Similarly, for $P = 10$, the number of parameters to be specified in Eq. (33) is 1031 and the search space dimensionality is 2^{22}.

This part of the learning is treated as being dynamic and off-line. Learning is dynamic as all the processes following fuzzy parameterization are affected by a change in this module. For instance, the FMAs change dynamically with changes in FCGs and so does the system's performance. Once the required performance specifications have been met, the system is ready for recognition at its best performance, satisfying certain conditions and requires no more learning. In the section that follows, we experiment with a set of possible GA strings and analyze the resulting performance of ORS.

6. Results and conclusion

Two experiments are tried with GA-based learning:

- The number of partitions is fixed and set to $P = 5$.
- Varying $5 \leqslant P \leqslant 10$.

6.1. Learning with fixed number of partitions

Table 3 shows the performance of ORS with the learning mechanism using a fixed number of partitions. The learning is carried out in 3000 generations. The training is performed individually for validation and generalization performance. The genetic structure having the best fitness function is chosen and corresponding rejection results are obtained. The recognition performance on trained data has improved to 96% while rejection performance is the same as

Table 3
GA-based performance of chess model base with fixed number of partitions

Validation performance		Generalisation performance					
Recognition %	Rejection %	Model	Recognition %	Rejection %			
96	84	1	88	83			
		2	91	84			
		3	84	84			
		4	79	84			
		5	76	85			
Validation performance with noisy test objects							
SNR	1.0	1.5	10	50	100	200	Mixed
Recognition %	40	60	40	40	80	40	56

the original ORS. With partial model learning, the results show a great improvement in recognition and a slightly higher improvement in rejection performance. Validation performance on noisy data shows enhancement in recognition. However, just as in the case of the original ORS, there is no direct relationship in the variation of performance with increase in noise levels in the test objects. Figs. 6–8 show the trend of the various design parameters during learning. For instance, the recognition performance shown in Fig. 6 shows a consistency in improved performance in relation to the start of the learning

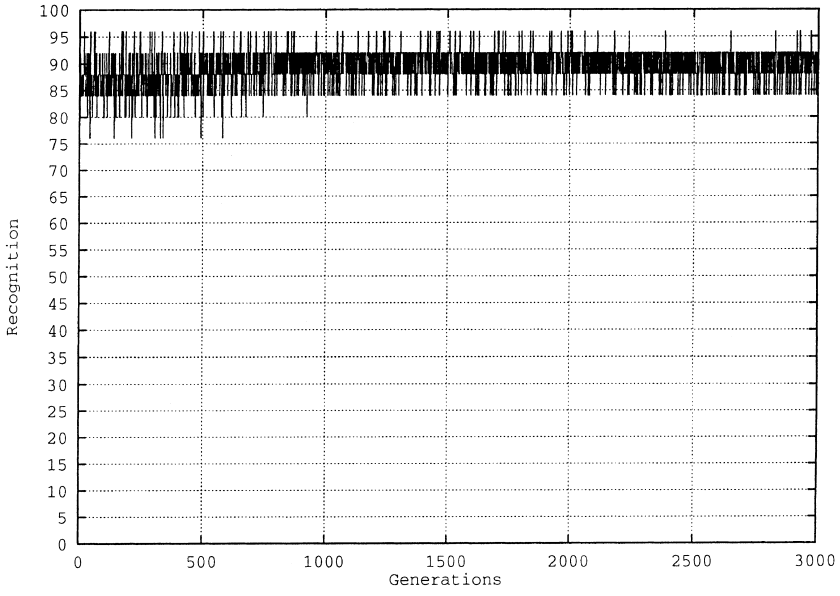

Fig. 6. Performance trend with fixed number of fuzzy partitions.

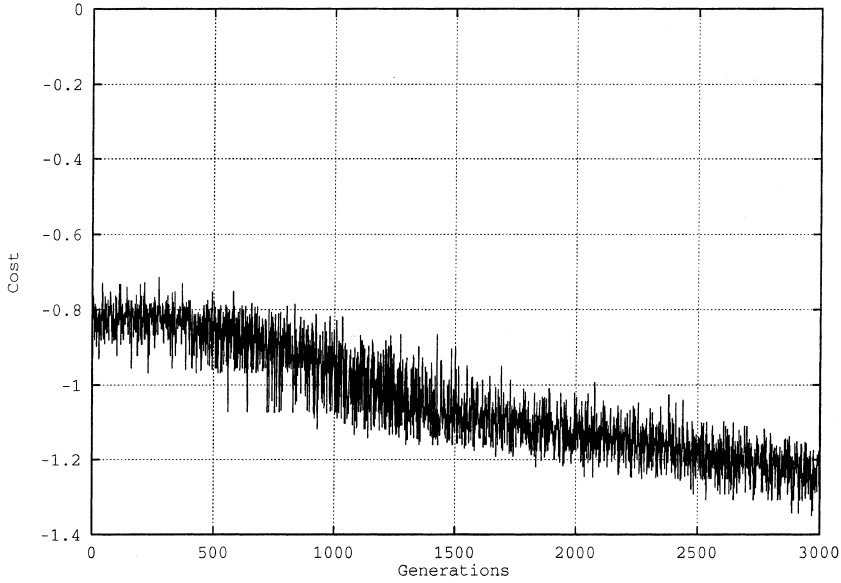

Fig. 7. Trend of cost function with fixed number of fuzzy partitions.

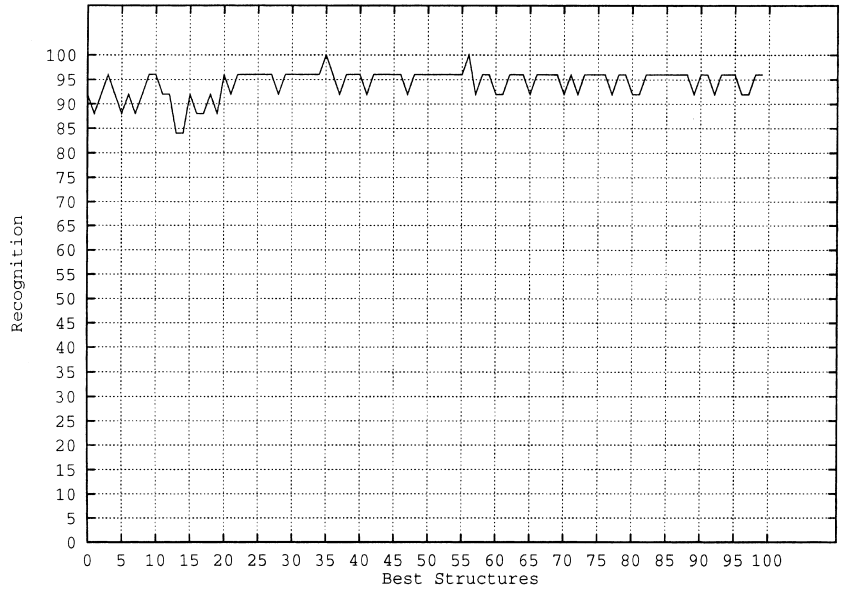

Fig. 8. Best performance with fixed number of fuzzy partitions.

process. In Fig. 7, *Cost*, which is the value of the objective function, tends to minimize over growing generations. These properties are attained because of the Elitist approach that retains the best structure. Fig. 8 shows the same

Table 4
GA-based performance of chess model base with varying number of partitions

Validation performance								
Recognition %	Rejection %							
100	100							
Validation performance with noisy test objects								
SNR		1.0	1.5	10	50	100	200	Mixed
Recognition (%)		40	40	40	60	60	60	50

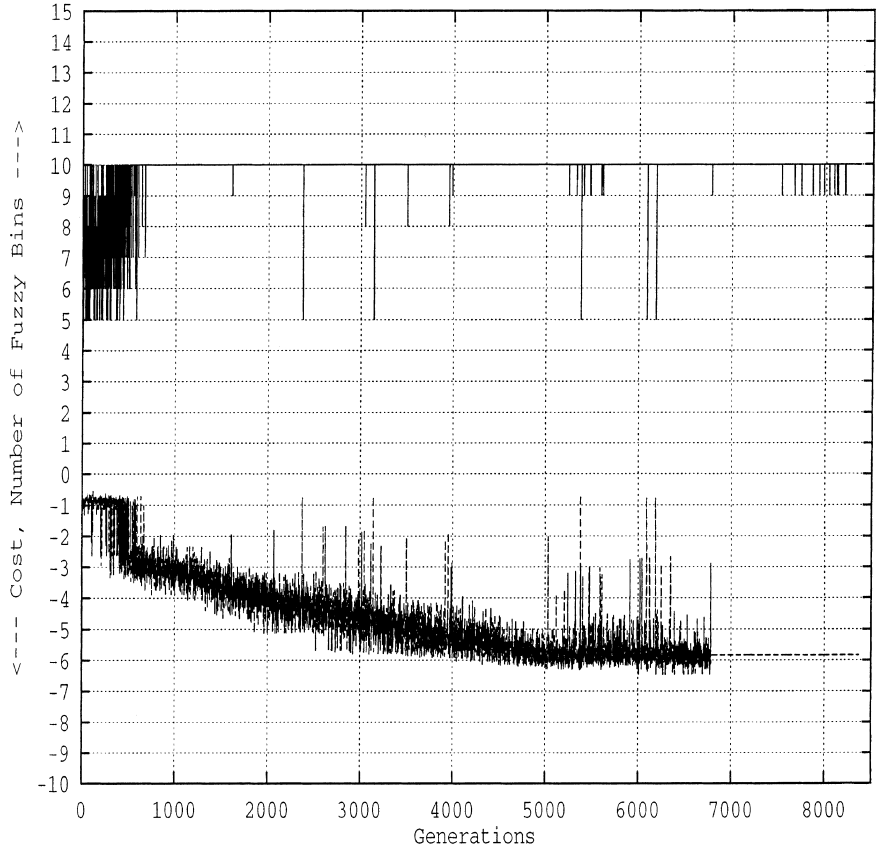

Fig. 9. Trend analysis of variable fuzzy partitions (upper plot) and cost (lower plot).

performance obtained by 100 best structures during the entire learning process. Although a 100% recognition efficiency is obtained in a couple of generations, most of them produce a 92–96% recognition efficiency. The variation in performance is in steps of 4%, since the number of objects in the database is 25.

6.2. Learning with varying number of partitions

Table 4 shows the performance of ORS due to the learning mechanism with varying number of partitions. We can see that both recognition and rejection performance on trained data have reached 100%. This result is achieved when the number of fuzzy partitions is set to 10. This is seen in the trend analysis of the number of fuzzy partitions allocated during each generation as shown in Figs. 9 and 10. Validation performance on noisy data shows a linear relationship between performance and noise levels. A moderate performance of

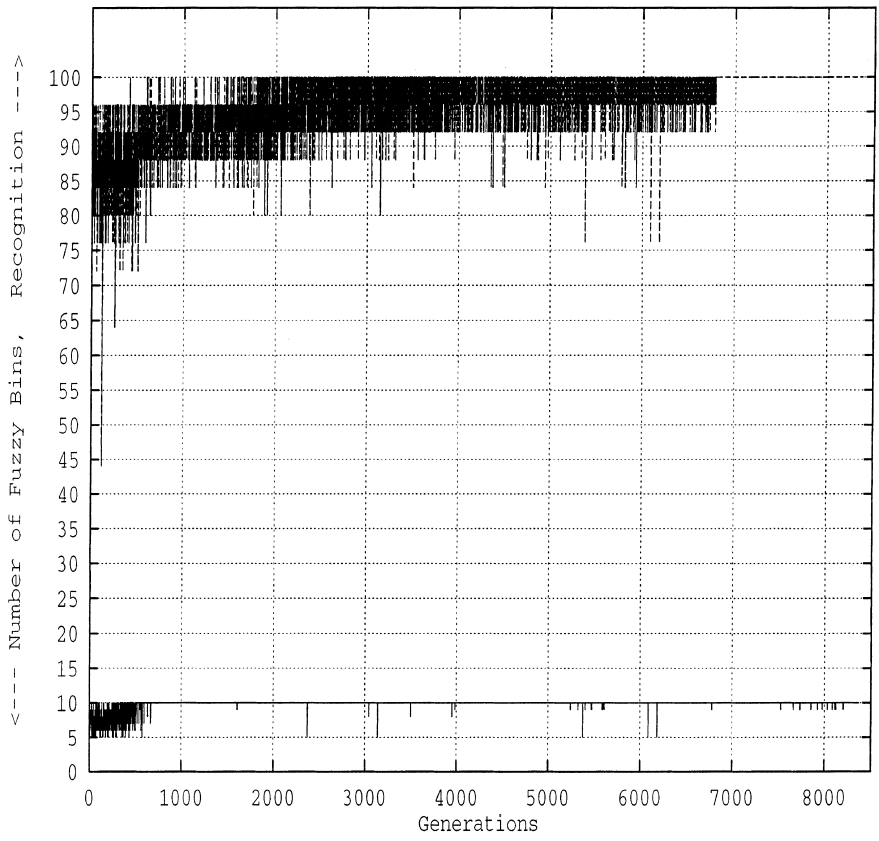

Fig. 10. Trend analysis of variable fuzzy partitions (lower plot) and performance (upper plot).

60% is maintained for lower noise levels, while at higher noise levels, it is maintained at 40%. Fig. 11 shows the variation of recognition performance and *Cost* as a function of P attained during each generation.

It is observed that fixing the structure length to be a constant for variable P can affect the crossover process for small structure lengths. It is possible that two structures may appear physically different but remain functionally the same based on the value of P and the crossover point. If the crossover points are further away from the structure length corresponding to P, then the result of crossover does not produce an offspring different from its parent. If this is true, the system should tend to choose gradual increase in the value of P over the many generations. But, the system favors values of $P = \{5, 9, 10\}$ as seen in Fig. 11. Hence no firm decision can be made at this stage on using fixed length structures to represent variable parameters.

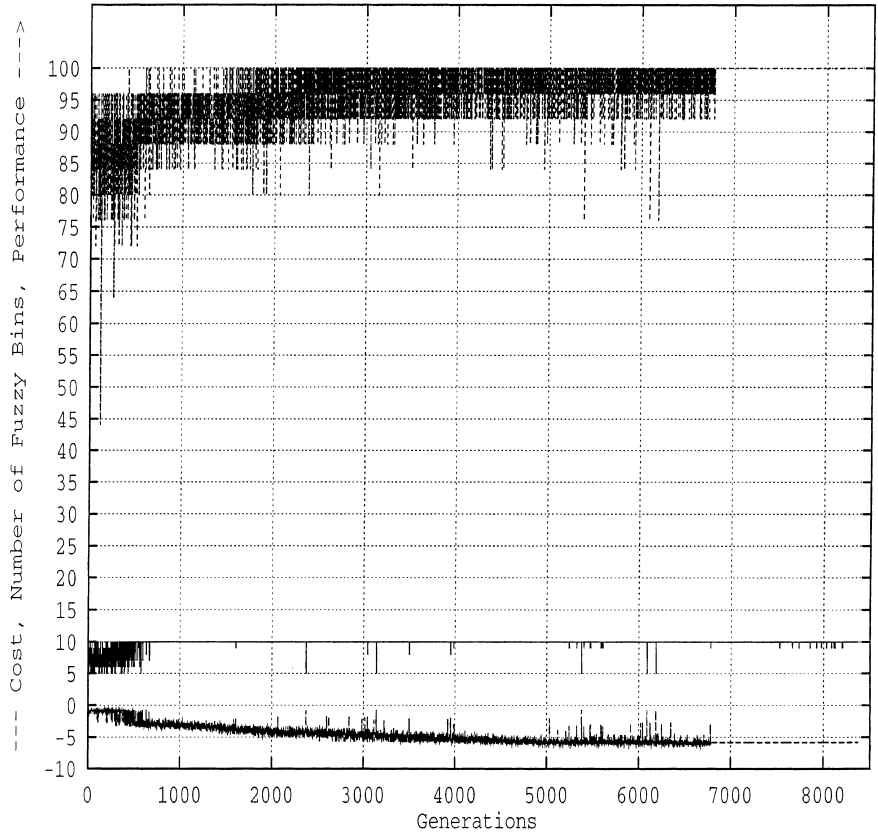

Fig. 11. Trend analysis of cost (lower plot), variable fuzzy partitions (middle plot) and recognition performance (upper plot).

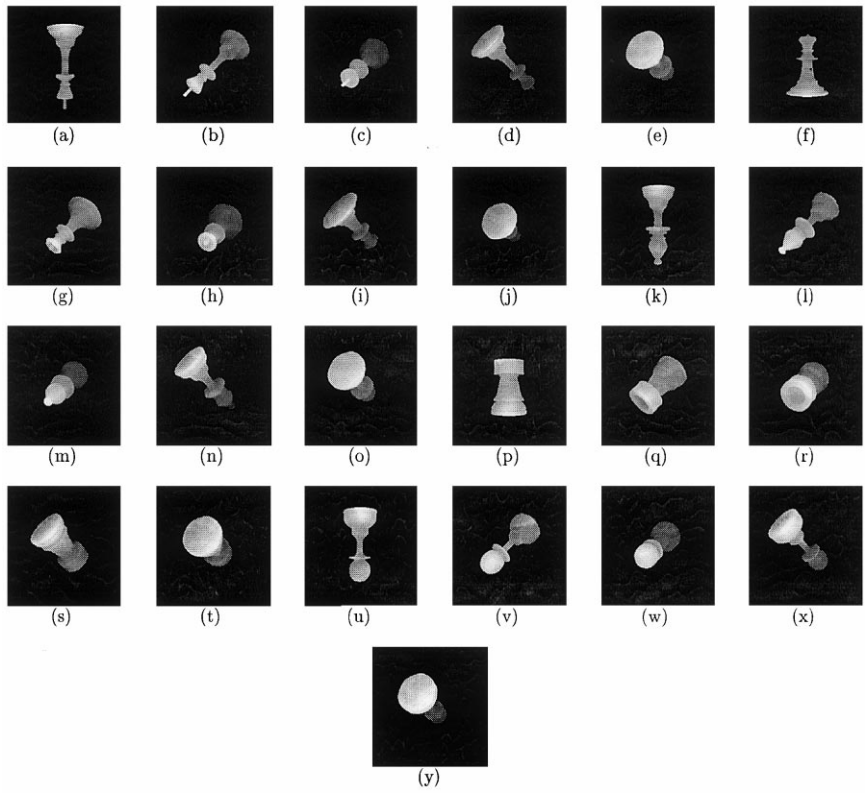

Fig. 12. Object instances for model construction: (a)–(e): king; (f)–(j): queen; (k)–(o): bishop; (p)–(t): rook; (u)–(y): pawn.

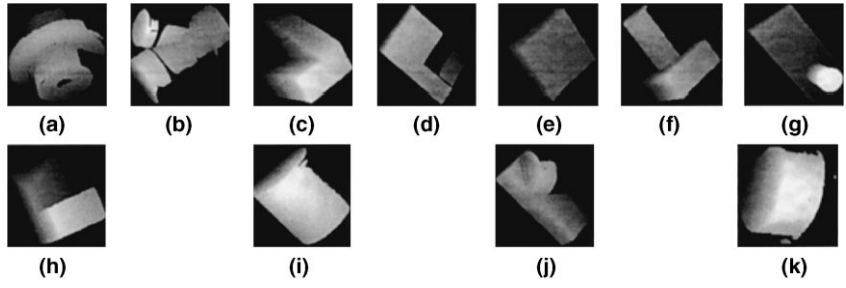

Fig. 13. Rejection hypothesis: object base.

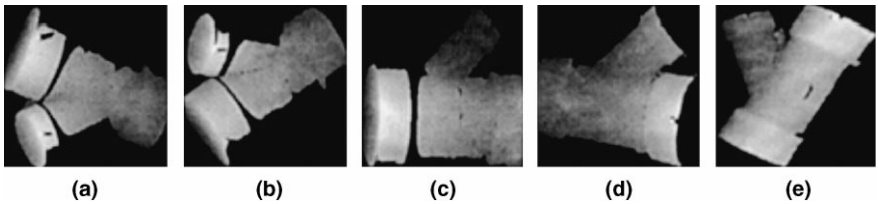

Fig. 14. Rejection hypothesis: instances of a class.

7. Conclusion

In this paper, we have proposed an off-line dynamic GA-based learning mechanism for a model based object recognition system that minimizes misclassification during recognition. The learning module is incorporated between the output and the feature fuzzification modules of ORS. The feature fuzzification process described in terms of the number of fuzzy membership functions, the extent of their overlaps and their shapes, adapts itself dynamically based on the performance of the system. An optimal set of parameters for this module is attained when the system shows a steady state optimal response. Once this performance is reached, learning is terminated. The result provides an improved performance measure under validation, generalization, rejection, and noisy test conditions. It is established that the learning process indeed enhances recognition. Performance can further be enhanced under noisy test conditions by modeling the noise distribution and training the system under this situation. We are currently investigate this aspect and develop a more robust model.

References

[1] M. Nagao, Control strategies in pattern analysis, Pattern Recognition 17 (1) (1984) 45–56.
[2] A.R. Rao, R. Jain, Knowledge representation and control in computer vision system, IEEE Expert (1988) 64–79.
[3] A. Rosenfeld, Image analysis: problems, progress and prospects, in: M.A. Fisch ler, O. Firschein (Eds.), Readings in Computer Vision, Morgan Kaufmann, Los Altos, CA, 1987.
[4] J.H. Holland, Adaptation in Natural and Artificial Systems, Michigan Press, MI University, 1975.
[5] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[6] C.L. Karr, E.J. Gentry, Fuzzy control of ph using genetic algorithms, IEEE Transaction on Fuzzy Systems 1 (1) (1993) 46–53.
[7] A. Homaifar, Ed. McCormick, Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms, IEEE Transaction on Fuzzy Systems 3 (2) (1995) 129–139.

[8] M. Valenzuela-Rendon, The fuzzy classifier system: A classifier system for continuously varying variables, in: Proceedings of the Fourth International Conference on Genetic Algorithms, San Diego, July 1991, pp. 346–353.
[9] Michael A. Zmuda, Louis A. Tamburino, Mateen M. Rizki, Generating pattern-recognition systems using evolutionary learning, IEEE Expert, August 1995, 63–68.
[10] P.K. Simpson, Fuzzy min–max neural networks part 1: classification, IEEE Transaction on Neural Networks 3 (5) (1992) 776–786.
[11] Shigeo Abe, Ming-Shong Lan, A method for fuzzy rules extraction directly from numerical data and its application to pattern classification, IEEE Transaction on Fuzzy Systems 1 (1992) 18–28.
[12] Naohisa Yamamoto, Hisao Ishibuchi, Ken Nozaki, Hideo Tanaka, Selecting fuzzy if-then rules for classification problems usinggenetic algorithms, IEEE Transaction on Fuzzy Systems 3 (3) (1995) 260–270.
[13] R. Fisher, The use of multiple measurements in taxonomic problems, Annals of Eugenics Part II 7 (1936) 179–188.
[14] R. Soodamani, Z.Q. Liu, Fuzzy surface descriptions for 3-D machine vision, in: Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics vol. 4/5, 22–25 October, Vancouver, British Columbia, Canada 1995, pp. 3238–3243.
[15] Terry Caelli, Ashley Dreier, Variations on the evidence-based object recognition theme, Pattern Recognition 27 (2) (1994) 185–204.
[16] B.P. Besl, R. Jain, Segmentation through symbolic surface descriptions, in: Computer Vision and Pattern Recognition, Miami, FL, USA, 1986, pp. 195–240.
[17] B.P. Besl, R. Jain, Invariant surface characteristics for 3d object recognition in range images, Computer Vision, Graphics and Image Processing 33 (1) (1986) 33–79.
[18] R. Soodamani, Z.Q. Liu, Fuzzy measures for surface segmentation, in: Fourth International Conference on Control, Automation, Robotics and Vision, ICARCV'96, vol. 2/3, Nanyang Technological University, Singapore, 4–6 December 1996, pp. 1363–1367.
[19] A.K. Jain, Fundamentals of Digital Processing, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.
[20] Bernard Moulin, Guy W. Mineau, John F. Sowa (Eds.), Conceptual Graphs for Knowledge Representation, Springer, Berlin, ICCS, 1993.
[21] S.M. Mohiddin, S. Ramalingam, Z.Q. Liu, Fuzzy Conceptual Graphs for Machine Vision, World Scientific, Singapore, 1994.
[22] George J. Klir, Tina A. Folger, Fuzzy Sets, Uncertainty, and Information, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.
[23] A.D. Bethke, Genetic algorithms as function optimizers, Ph.D. thesis, Department of Computer and Communication Sciences, University of Michigan, 1981.
[24] K.A. DeJong, Analysis of the behaviour of a class of genetic adaptive systems, Ph.D. thesis, Department of Computer and Communication Sciences, University of Michigan, 1975.
[25] D.R. Frantz, Non-linearities in genetic adaptive search, Ph.D. thesis, Department of Computer and Communication Sciences, University of Michigan, 1972.
[26] R.B. Hollstien, Artificial genetic adaptation in computer control systems, Ph.D. thesis, Department of Computer and Communication Sciences, University of Michigan, 1971.
[27] A. Brindle, Genetic algorithms for function optimization, Ph.D. thesis, Computing Science Department, University of Alberta, 1981.
[28] S.F. Smith, Flexible learning of problem solving heuristics through adaptive search, in: Proceedings of the Eighth International Joint Conference on Intelligence (IJCAI), August 1983.
[29] James E. Baker, Reducing bias and inefficiency in the selection algorithm, in: Genetic Algorithms and Their Applications, LEA, Cambridge, MA, July 1987, pp. 14–21.